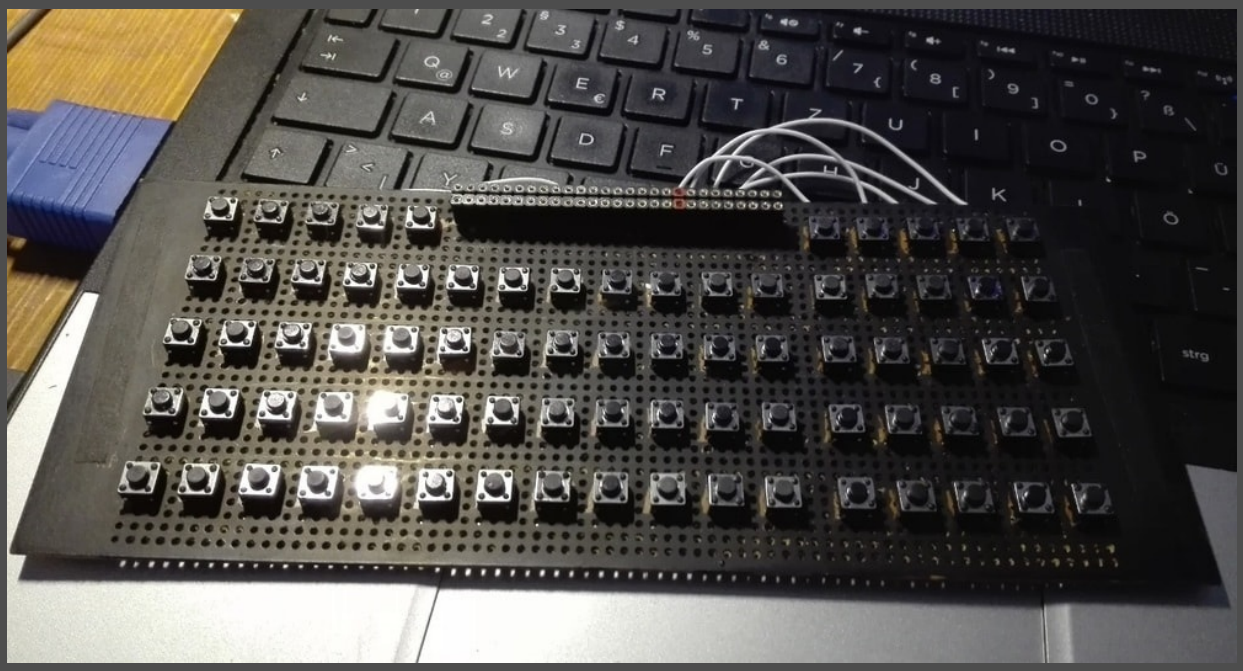
Public Domain 2010-2019 | voidnill



Cyberdeck

Preface

Keyboards are ubiquitous in our everyday life. As a computer keyboard, at ATMs, digitally on our smartphone and tablet or when we use our calculator. But if we take a closer look at the technology and functionality of a keyboard, we quickly get into a stutter. We have to work through the keyboard matrix, learn what a keyboard controller) is and how ghosting (see Dribin, 2000) can disrupt our development. In this section we will create a simple keyboard that will serve as a module for our single board computer keyboard. This should be created without a controller, since this is implemented in a different documentation.

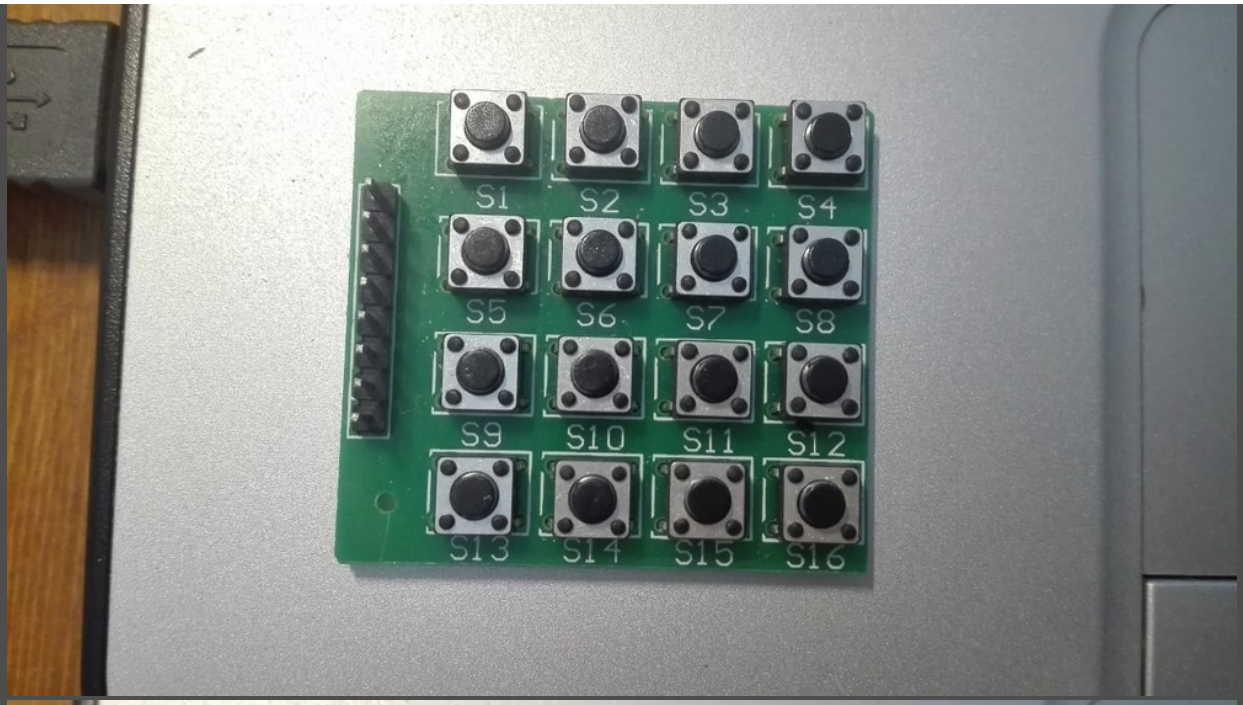
Although the project can also be implemented without the in-house development of a computer keyboard, this means that you learn less about the technology you use. Putting a Teensy on a few mechanical buttons leads to quick results but is more important for hobby development. At the beginning we will also use external modules (keyboard controller), but in the course of time we will gradually rebuild and replace them, even if we may have to do without convenient additional functions as a result. No mechanical buttons have been used, as they have no reasonable price/performance ratio (€8.30/pc.).

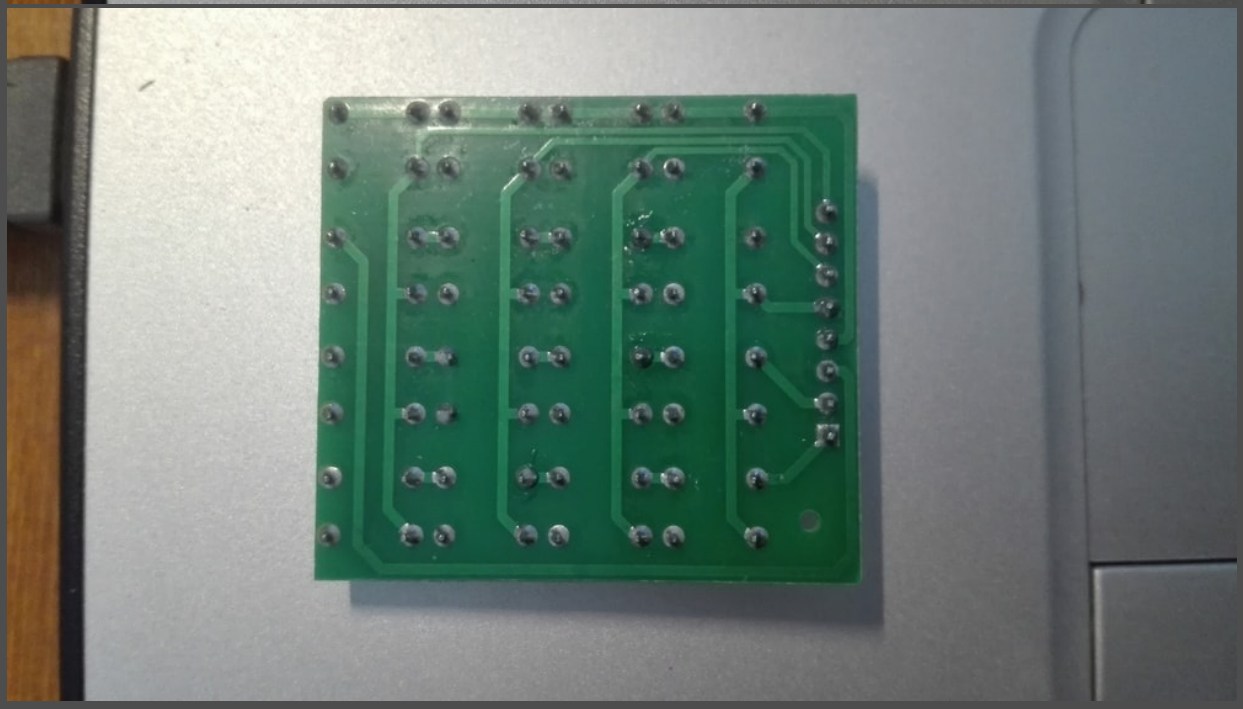
Problem definition

The task can be described in a few points. The keyboard must be designed so that it can be integrated as a module into existing projects. It must therefore have sufficient hardware interfaces that can also be used universally. Different standards should be waived as best as possible. The keyboard must be developed cost-effectively, since it is only a prototype in the development stage and should progress as quickly as possible in the development process. Since the prototype is a functional object, it must be manufactured with sufficient stability, design must be dispensed with and, if at all, it should be functional.

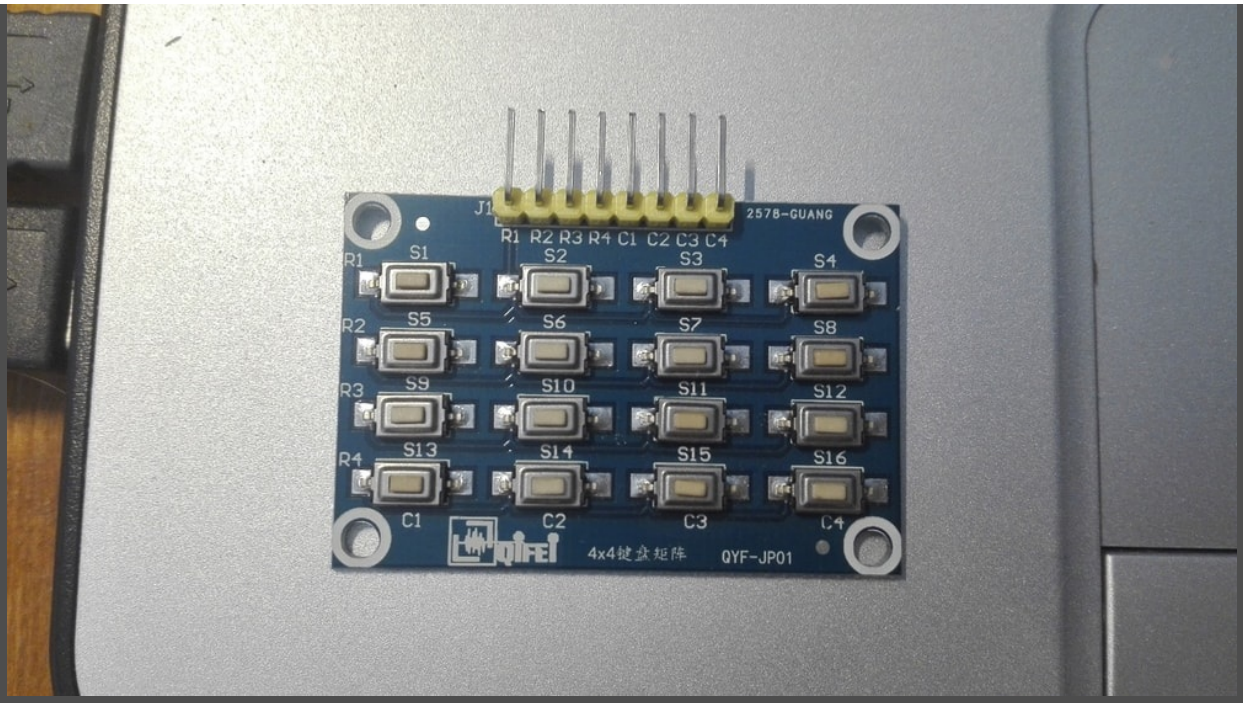
Research

In order to implement our project sensibly, we do not need much information material. How a keyboard generally works is completely sufficient. Once you have worked out the theoretical concept, the implementation in practice is only the execution of individual steps. A good book for beginners is MAKE: Electronik: Lernen durch Entdecken, which was used as a source in the project documentation.





The keyboard shown above is a typical generic brand product from a wholesaler in China and was used for research. On the solder side you can see the traces for a board with a single layer. This information will be included in the project later on and partially copied. Below another keyboard ordered to examine the two-contact push buttons.

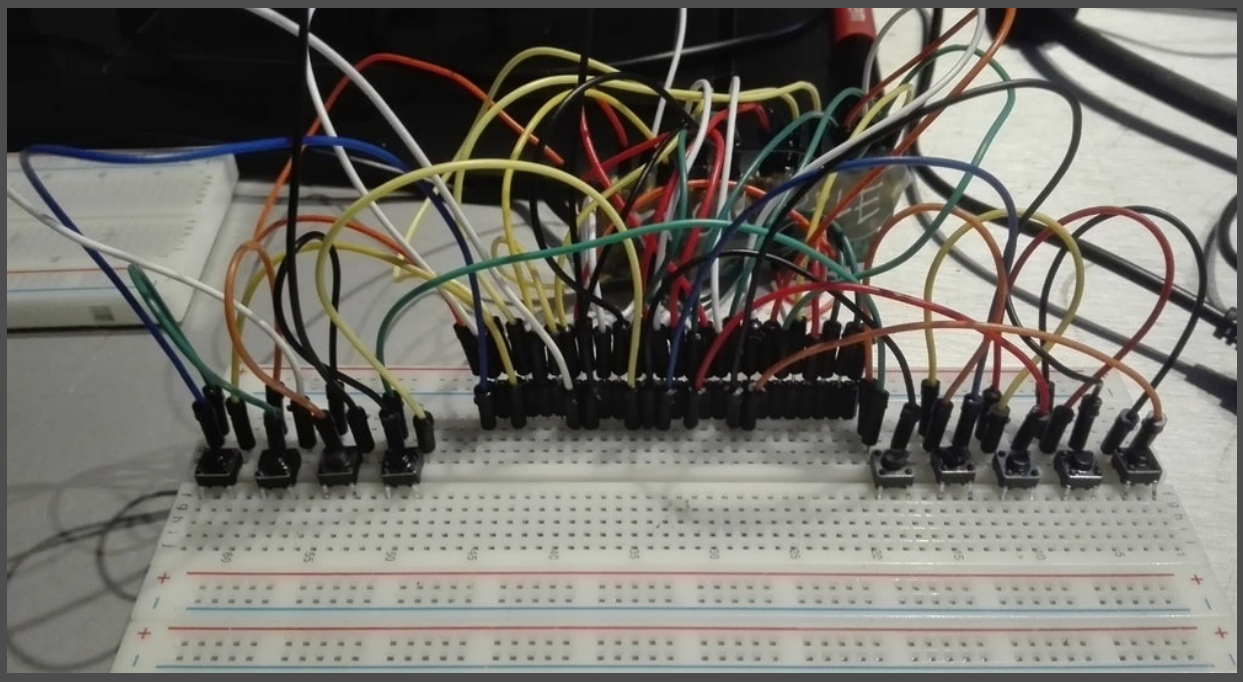


Material

As material we need at least 100 mini push buttons, thin cable for the connections between the push buttons. A large breadboard on which all components have to fit. Pin headers (4×10). A keyboard controller with USB port, or the whole keyboard. One should take an old one, because the leftovers are no longer usable without the controller. There are also PS/2 connectors, but we will not use them here. Several breadboards and at least 200 connector cables.

Realisation

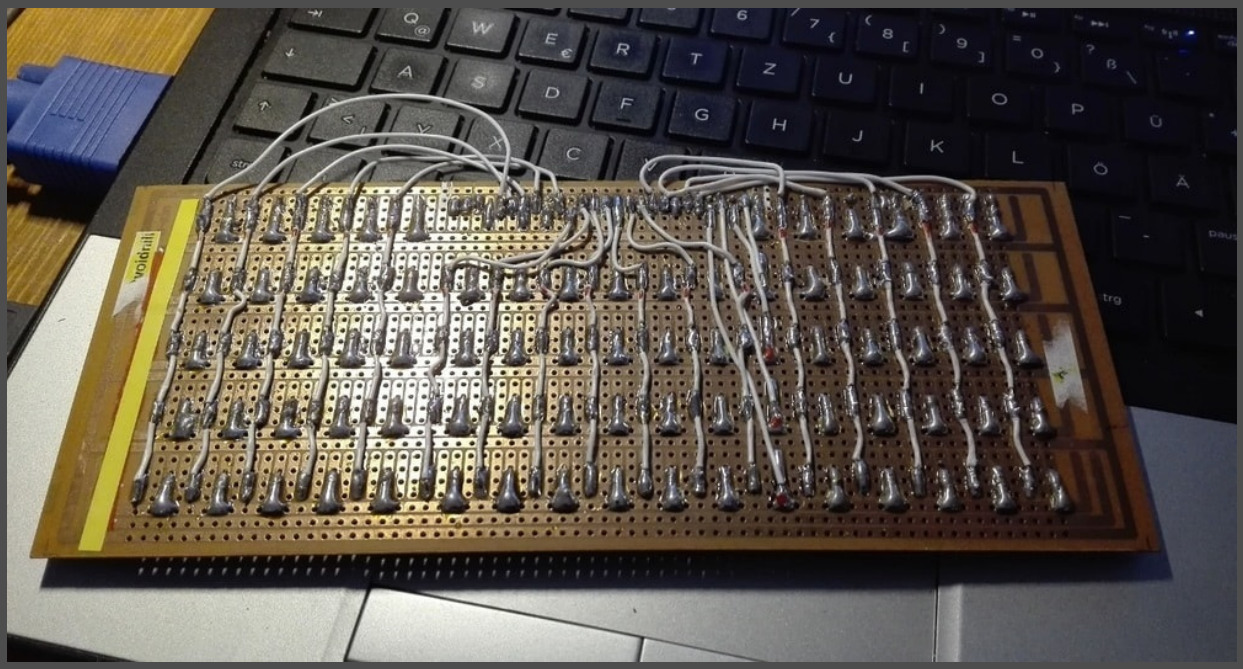
In order to become familiar with the subject, some tutorials (like: jacking a usb keyboard) were worked through and implemented. The keyboard controller was connected to the computer via USB and the signals were read out via command line. With the theoretical material the concept of the keyboard matrix could be better understood.



We build the circuit series by series. Two connector cables must always terminate on one push-button. Since each keyboard controller has been programmed differently, I cannot give an exact description of how to connect the cables here. Simply plug a cable into the first contact of the controller and connect it to the print button. Now take a second cable and connect it to the second contact of the controller. Press the push button once and see what happens in the command line. If nothing happens, you go one contact to the right. If a letter (or number) appears, write it down and go to the next button. So you try them all until you have at least the numbers from 0-9 and the complete alphabet together.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	^	F9		0x0	AD	0x0	Entf	6	Bild auf	5	Einfg	Pos 1	F8	F2	F1	Firefox öffnen	Strg	
2	1	F10			8	0	Ruhe Modus	7	Bild ab	4	Ruhe Modus	Ende	9	3	2		F5	0x0
3	Y	Enter	0x0	0x0	,	#	Num	M	Num Pad *	V	Num Pad /	Win Media Player	.	C	X		Strg	0x0
4	0x0	F12	0x0	Alt	0x0	-	Unten	N	Num Pad -	B	Rechts	Links	Kontext Menü	0x0	0x0	Taschen Rechner		0x0
5	Esc	F11	0x0	Alt		Signal Ton	Space	H	Entf	G	Einfg	Oben		F4	<		F15	Einfg
6	A	#	Umschalt	0x0	K	0x0	Ende	J	Bild ab	F	Unten	Enter	L	D	S	Win	F14	0x0
7	Tab	Rück Taste	Umschalt	0x0	+	0x0	Links	Z	Rechts	T	Löschen	Löschen	F7	F3	Fest Stell Taste	0x0	F13	Win
8	Q	0x0		Rollen	I	P	Pos 1	U	Bild auf	R	Oben	Num Pad +	O	E	W	Einfg	Pause	0x0

After testing the first board, the keyboard was designed as it will be used in the later design. The soldering and component side and individual fixtures are shown below. As you can see from the image material, the connections on the trace are kept as simple as possible, so that it is still logically traceable after days. I have soldered on two female pinheaders which are not necessary, but can be an advantage especially in the development. As a rule, it is better to create as many slots as possible.



After completion of the keyboard, the controller is implemented in the next step to develop a communication between input device and computer. The focus will be on the next section of the microcontroller, which ultimately turns passive hardware into a finished input device. A further optimization of the keyboard is also aimed for.